PATENT ABSTRACTS OF JAPAN

(11)Publication number : 07-281897
(43)Date of publication of application : 27.10.1995

(51)Int.Cl. G06F 9/42
G06F 9/46

(21)Application number : 07-108056
(22)Date of filing : 06.04.1995

(71)Applicant : HEWLETT PACKARD CO <HP>
HITACHI LTD
(72)Inventor : AMERSON FREDERIC C
ENGLISH ROBERT M
GUPTA RAJIV
WATANABE HIROSHI

(30)Priority
Priority number : 94 223804 Priority date : 06.04.1994 Priority country : US

(54) REGISTER ASSIGNMENT METHOD AND REGISTER FILE PORT ACCESS DEVICE

(57)Abstract:
PURPOSE: To reduce the amount of save and restore of a register at the time of a procedure call/return.
CONSTITUTION: Only registers of a truly required number of registers assigned from a register pool for each procedure by improving the register window system. This assignment is made by a called procedure itself independently of a caller procedure. When the registers in the register pool are used out, part of contents of the registers assigned already is saved in a memory to make an idle register.

LEGAL STATUS

[Date of request for examination]	19.03.2002
[Date of sending the examiner's decision of rejection]	
[Kind of final disposal of application other than the examiner's decision of rejection or application converted registration]	
[Date of final disposal for application]	
[Patent number]	3529888
[Date of registration]	05.03.2004
[Number of appeal against examiner's decision of rejection]	
[Date of requesting appeal against examiner's decision of rejection]	
[Date of extinction of right]	

(19)日本国特許庁（ＪＰ）　(12)公開特許公報（Ａ）　(11)特許出願公開番号

特開平7－281897

(43)公開日　平成７年(1995)10月27日

(51)Int.Cl.[6]		識別記号	庁内整理番号	ＦＩ	技術表示箇所
Ｇ０６Ｆ	9/42	３３０　Ｒ			
	9/46	３１３　Ｂ	7737－5Ｂ		

審査請求　未請求　請求項の数10　ＦＤ　（全 15 頁）

(21)出願番号　特願平7－108056

(22)出願日　平成７年(1995)４月６日

(31)優先権主張番号　２２３，８０４

(32)優先日　1994年４月６日

(33)優先権主張国　米国（ＵＳ）

(71)出願人　590000400
ヒューレット・パッカード・カンパニー
アメリカ合衆国カリフォルニア州パロアルト　ハノーバー・ストリート　3000

(71)出願人　000005108
株式会社日立製作所
東京都千代田区神田駿河台四丁目６番地

(72)発明者　フレドリック・シー・アマーソン
アメリカ合衆国カリフォルニア州サンタ・クララ、ヒルズデイル・アベニュー　451　アパートメント・ナンバー・ピー

(74)代理人　弁理士　上野　英夫

最終頁に続く

(54)【発明の名称】　レジスタ割当て方法及びレジスタファイルポートアクセス装置

(57)【要約】
【目的】プロシージャコール、リターンの際のレジスタのセーブ、リストアの量を減らす。
【構成】レジスタウインドウ方式を改良して、レジスタプールからプロシージャ毎に真に必要な個数のレジスタだけを割り当てる。この割り当ては、呼び出されたプロシージャが呼出し元のプロシージャとは独立に自分で割り当てを行う。レジスタプール中のレジスタを使い切ったら、既に割り当てられているレジスタの内容の一部をメモリにセーブして、空きを作る。

1

【特許請求の範囲】

【請求項１】物理レジスタセットを有するディジタルコンピュータにおいて、コンパイラの介在なしに動的にレジスタをプロシージャに割り当てる方法であって、

複数のスタックレジスタを有する論理レジスタスタック（５８）を定義するステップと、

前記論理レジスタスタックを前記物理レジスタセット（図４Ａ）にマッピングするためのオフセットを定義するためにローカルリローケション項（ｌｒｅｌ）を初期化するステップと、

第１のスタックポインタ値（ＴＯＬ）を初期化することにより、第１のプロシージャにより指定された任意の個数のスタックレジスタ（６２）をローカルレジスタとして第１のプロシージャ（Ａ）に割り当てて、前記論理レジスタスタック内の前記ローカルレジスタ（６２）を区切るステップと、

前記第１のプロシージャ実行の間のレジスタアクセス操作に関連して、前記論理リロケーション項（ｌｒｅｌ）に応答して各ローカルレジスタ論理アドレス（Ｒ、図５）を前記物理レジスタセット（ｒ）にマッピングするステップを設けたことを特徴とするレジスタ割当て方法。

【請求項２】前記第１のスタックポインタ値（ＶＯＬ）をストアして第２のスタックポインタ値（ＯＴＯＬ）を形成するステップと、

前記第１のスタックポインタ値（ＴＯＬ）をインクリメントすることにより、前記第１のプロシージャにより指定された任意の個数の追加のスタックレジスタ（６４）をパラメータ引渡レジスタとして前記第１のプロシージャ（Ａ）に割り当て、パラメータ引渡レジスタを含ませるようにするステップと、

呼び出されたプロシージャ（Ｂ）が参照するために、選択されたパラメータを前記割り当てられたパラメータ引渡レジスタ（６４）にストアするステップであって、前記ストアするステップがパラメータ引渡レジスタを前記ローカルリロケーション項（図５）に応答して前記物理レジスタセットにマッピングするステップを含むようなステップを設けたことを特徴とする請求項１記載のレジスタ割当て方法。

【請求項３】第２のプロシージャ（Ｂ）を呼び出し、第１のプロシージャパラメータ引渡レジスタ（６４）を有する初期ローカルレジスタ空間を前記第２のプロシージャに割り当て、これにより前記レジスタにストアされた前記選択されたパラメータを、メモリ参照なしに前記第２のプロシージャに利用可能にするステップと、

前記スタックポインタ値（ＴＯＬ）をインクリメントすることにより、前記第２のプロシージャにより指定される任意の個数の追加のスタックレジスタをローカルレジスタ（６６）として前記第２のプロシージャ（Ａ）に割り当てて、前記第１プロシージャのローカルレジスタの

2

内容をはじめにメモリにセーブすることなしに、前記第２のプロシージャのローカルレジスタ（６６）を含ませるようにするステップと、、前記第２のプロシージャから復帰したとき、前記スタックポインタ値（ＴＯＬ）をローカルレジスタ（６６）の個数だけデクリメントすることにより前記ローカルレジスタを割り当て解除して、ローカルレジスタ内容をセーブ及び復元することなく前記第２のプロシージャを呼び出しまたそこから復帰するステップを設けたことを特徴とする請求項２記載のレジスタ割当て方法。

【請求項４】前記第２のプロシージャを呼び出したとき前記第１及び第２のスタックポインタ値（ＴＯＬ，ＯＴＯＬ）をストアして、前記第２のプロシージャから復帰した際に参照するためのストアされた値（制御レジスタＡ）を形成するステップを設け、

ここにおいて、前記割り当て解除するステップが前記第１及び第２のスタックポインタ値を前記ストアされた値にリセットするステップを含むことを特徴とする請求項３記載のレジスタ割当て方法。

【請求項５】追加のパラメータレジスタを前記第１のプロシージャに割り当てるステップを設け、

ここにおいて、前記ストアするステップが、前記第２のプロシージャからの復帰の際に参照するために前記追加のパラメータレジスタにスタックポインタオフセット値をストアするステップを含むことを特徴とする請求項２，３または４記載のレジスタ割当て方法。

【請求項６】ローカル下限（ＢＯＬ）ポインタ値を初期化して、現プロシージャに割り当てられた一連のスタックレジスタの一方の端を指示し、前記現プロシージャに割り当てられた一連のスタックレジスタの他方の端が前記第１のスタックポインタ値（ＴＯＬ）により指示され、

ここにおいて、前記ローカルリロケーション項（ｌｒｅｌ）が、前記ローカル下限ポインタ値から予め定められた定数個のスタックレジスタを引いたものに等しいことを特徴とする請求項２，３または４記載のレジスタ割当て方法。

【請求項７】前記レジスタスタックを介してアクセス可能なソフトウエアスタックの深さを指示するために有効範囲下限ポインタ（ＢＯＶ）を初期化するステップを含み、

ここで前記第１のスタックポインタ値（ＴＯＬ）をインクリメントすることがモジューロ加算及びモジューロ物理レジスタ数を使用して行われ、それにより前記レジスタセットがリングとして管理されるようにし、更に、

前記第１のスタックポインタのインクリメントが有効範囲下限ポインタ（ＢＯＶ）値より大きい値を招くとき、レジスタオーバーフロー状態を指示するステップを含むことを特徴とする請求項２，３または４に記載の方法。

【請求項８】第１の循環レジスタポインタ値（ＢＯＲ）

及び第２の循環レジスタポインタ値（ＴＯＲ）を前記第１のスタックポインタ値（ＴＯＬ）に初期化するステップと、

前記第２の循環レジスタポインタ値（ＴＯＲ）及び前記第１のスタックポインタ値（ＴＯＬ）を、前記呼び出されたプロシージャにより循環レジスタとして指定されたレジスタの任意の個数だけインクリメントすることにより、レジスタを呼び出されたプロシージャ（Ｃ，図３Ｆ）に循環レジスタ（７４）として割り当てるステップと、

前記呼び出されたプロシージャから復帰するに先立って、前記第２の循環レジスタポインタ値及び前記第１スタックポインタ値を循環レジスタの個数だけデクリメントすることにより、前記循環レジスタの割当てを解除するステップを設けたことを特徴とする請求項１記載のレジスタ割当て方法。

【請求項９】請求項１に記載の方法を実現するためのレジスタファイルポートをアクセスする物理アドレスを提供するためのレジスタファイルポートアクセス装置（１４０）において、

現プロシージャから仮想アドレス（Ｒ）を受け取る入力手段（１４２）と、

前記仮想アドレス（Ｒ）を予め設定された定数と比較して、前記仮想アドレスがスタティックレジスタを指示するのかそれともスタックレジスタを指示するのかを判定する比較手段（１５０）と、

前記仮想アドレスをローカルリロケーション項（ｌｒｅｌ）に加算して、第１の物理アドレスを形成する手段（１５４）と、

前記仮想アドレス（Ｒ）と前記第１の物理アドレスの何れか一方を選択し、選択されたアドレス（ｒ）を前記レジスタファイルポートに結合するマルチプレクサ手段（１４６）と、

前記マルチプレクサ手段に結合され、前記仮想アドレス（Ｒ）がスタックレジスタを指示している場合には前記第１の物理アドスを選択し、前記仮想アドレスがスタティックレジスタを指示している場合には前記仮想アドレスを選択し、これによりスタックレジスタへの参照を前記現プロシージャに割り当てられた物理レジスタアドレスに振り替える前制御手段（１５２）を設けたことを特徴とするレジスタファイルポートアクセス装置。

【請求項１０】前記仮想アドレス（Ｒ）と第１及び第２の循環レジスタポインタ値（ＢＯＲ，ＴＯＲ）とを比較して、前記仮想アドレスが前記現プロシージャに循環レジスタとして割り当てられたレジスタを指示するかどうかを判定する比較手段（１６４，１６６）と、

第１の物理レジスタを形成するために前記仮想アドレス（Ｒ）を第１の循環リロケーション項（ｒｒｅｌ）に加算する手段（１７０）と、

前記仮想アドレス（Ｒ）を第２の循環リロケーション項

（ｒｒｅｌ＃）に加算して、第２の物理レジスタを形成する手段（１７２）と、

前記第１及び第２の物理アドレスの何れか一方を選択し、選択されたアドレス（ｒ）を前記レジスタファイルポートアドレス端子（１４４）に結合するマルチプレクサ手段（１６２）と、

前記マルチプレクサ手段（１６２）を制御して、前記仮想アドレス（Ｒ）が前記循環レジスタセット内での回り込み（wraparound）を起こさない場合には前記第１の物理アドスを選択させ、前記仮想アドレスが前記循環レジスタセット内で回り込みを起こす場合には前記第２の物理アドレスを選択させる制御手段（１６８，１５２）を設け、

前記第１の循環リロケーション項（ｒｒｅｌ）は前記ローカルリロケーション項（ｌｒｅｌ）に循環レジスタベース値（ＲＲＢ）を加算した結果に等しく、前記第２の循環リロケーション項（ｒｒｅｌ＃）は前記ローカルリロケーション項（ｌｒｅｌ）に前記循環レジスタベース値（ＲＲＢ）から前記循環レジスタセットのサイズを差し引いたものを加算した結果に等しく、これにより前記循環レジスタセット内での前記回り込みを調整することを特徴とする請求項９記載のレジスタファイルポートアクセス装置。

【発明の詳細な説明】

【０００１】

【産業上の利用分野】本発明は一般にディジタルコンピュータに関し、特にディジタルコンピュータのレジスタの管理に関する。

【０００２】

【従来技術及びその問題点】ノイマン型ディジタルコンピュータは操作中に種々の値を保持するためのレジスタセットを備えている。レジスタセットの大きさは可変である。全てのノイマン型装置は少なくともプログラムカウンタ（ＰＣ）を備えている。一般に、オペランド及び結果を保持するためのレジスタ（“演算レジスタ”）も備えている。ＲＩＳＣ（縮小命令セットコンピュータ）マシンは、一般に、メモリへの書込み・メモリからの読出しを行うがデータに対する操作は行わないロード（ＬＯＡＤ）及びストア（ＳＴＯＲＥ）命令以外では、レジスタからレジスタへの命令（メモリに直接アクセスする命令から区別される）しか持っていない。ＲＩＳＣマシンは大きなレジスタセットを持つ傾向があり、例えば３２個またはもっと多くにも達する。レジスタは、中間結果、アドレスインデックス及びサブルーチンのような呼び出し元と呼び出し先のプロシージャ間の引渡データ（パラメータ）を保持するために使用される。幾つかのプロセッサは汎用レジスタに加えて浮動小数点レジスタを備えている。ＣＩＳＣアーキテクチャは、通常、評価スタックを備えており、評価スタックはオペランドが明示的には示されないＯアドレス演算を提供する。ＲＩＳ

Cは、通常、評価スタックを備えていない。コンパイラは、ＲＩＳＣアーキテクチャにおいては、演算のためではなく、通常はパラメータ引渡及びレジスタ内容の追出し(register spills)のために、スタックをメモリ内に保持する。殆どのアーキテクチャにおいて、プロシージャ呼び出しに当たってのレジスタのセーブと復元のオーバーヘッドは負担になる。このオーバーヘッドはメインメモリ参照の５％から４０％を占める。このオーバーヘッドを減らすために、いくつかのバンクを設け、新たなバンクを各呼び出されたプロシージャに割り当てることが知られている。この技術はレジスタウインドウと称される。J. Hennessy, D. Patterson著のComputer Architecture — a Quantitative Approach(1990) Section 8. 7、を参照のこと。レジスタウインドウを使用することにより、これらのレジスタバンクつまり“ウインドウ”を、パラメータを引き渡すための共通エリアを提供するためにオーバーラップさせる。レジスタはプロシージャ呼出しに当たって変化しないグローバル(global)レジスタと、プロシージャ呼出しに当たって変化するローカルレジスタに分けられる。バッファが一杯になった後に、プロシージャ呼び出しが行われたとき（ウインドウオーバフロー）、またはバッファが空になった後にプロシージャからの復帰が起こったとき（ウインドウアンダーフロー）、レジスタのブロックはメモリにセーブされる。

【０００３】レジスタウインドウは、現在、サンマイクロシステムのＳＰＡＲＣ（登録商標）アーキテクチャで実現されており、更にレジスタウインドウをリング構成で操作することを開示している米国特許第5,159,680号で説明されている。米国特許第5,233,691号は、バス競合を起こさない時間にメモリにレジスタを予め書き込むことにより、オーバーフローによる書き込みを行う必要性を減らすレジスタウインドウシステムを開示している。オーバーラッピングウインドウを実現する高性能レジスタファイルは米国特許第5,226,142号及び米国特許第5,226,128号に開示されており、米国特許第5,083,267号及び米国特許第5,036,454号はループのために循環レジスタを使用することを開示している。

【０００４】レジスタウインドウのような従来技術のアーキテクチャが有する１つの問題は、レジスタのバンク（すなわちレジスタウインドウ）のサイズが固定されていることである。このサイズはプロシージャ毎に変化させることができない。その結果、プロシージャに割り当てられたローカルレジスタエリア内の全てのレジスタが実際にそのプロシージャにより使用されるわけではなく、逆に、多くの場合、プロシージャにそのプロシージャが必要とする十分なレジスタが割り当てられているわけではない。これにより、メモリ参照が最適化されないために、パフォーマンスの低下が引き起こされる。

【０００５】レジスタウインドウの他の制限は、オーバーラップレジスタの個数もまた固定されていることである。この場合にも、その個数は実際に呼び出されるプロシージャが必要とするパラメータの数を充分に上回ることがあり、一方ではレジスタの使用密度が減る。その上、この固定されたオーバーラップにより、単一のプロシージャ呼び出しに関して引渡されるパラメータの個数が勝手に限定される。

【０００６】循環レジスタ(rotating register)空間は、あるデータを使用する操作が呼び出される何サイクルか前にそのデータの準備を開始しそのデータが必要になったときにはそれが使用可能になっているようにするために、ソフトウエアのパイプライン化されたループによって使用される。ソフトウエアのパイプライン化されたループ内で必要とされるレジスタの個数は、ループの特性により変わる。もし、従来技術のように循環レジスタ空間のサイズが固定されているなら、大多数のループを扱うために例えば６４個のレジスタのような広大な空間を割り当てなければならない。しかし、１６個またはそれ以下のレジスタしか必要としない多くの小ループがあり、また６４個よりも多くのレジスタを必要とする多くの大ループがある。小ループの場合には、多くのレジスタが不必要に割り当てられまた開放され、大ループの場合には、レジスタの不足により処理速度が低下する。

【０００７】上記事情に鑑み、従来技術のレジスタウインドウのグループサイズが固定されているという枠組みに制約されない、レジスタの効率的な割り当て及び割り当て解除方法を提供することが必要とされる。

【０００８】

【目的】上記背景に鑑み、本発明の目的は、コンピュータにおいて呼び出し及び復帰操作の平均速度を向上させることにある。他の目的は、プロセッサの処理においてレジスタセーブ及び復元の回数を最小限にすることにある。他の目的は、呼び出されたプロシージャにより必要とされる一時的なローカル記憶装置を効果的に割り当てることにある。更に本発明の目的は、ルーチンの呼び出し元、あるいは呼出し先が使用する記憶装置にかかわりなく十分なレジスタ記憶装置を割り当てることにある。更に他の目的は、現在使われていないレジスタについてはセーブや復元を行わないことによって、効率を向上することにある。他の目的は、プロセッサ内で使用可能な限定された個数のレジスタを割り当て及びセーブすることに関連するオーバーヘッドをへらすことにある。更に他の目的は、プロシージャが必要とするならばレジスタの全範囲をそのプロシージャにより利用できるようにすることまで含む、呼び出されたプロシージャの要求を満たすために、レジスタセットを動的に分割することにある。他の目的は、ソフトウエアのパイプラン化されたループのためにプロシージャが必要とするのと丁度同じ個数の循環レジスタをそのプロシージャに割り当てることにある。他の目的は、レジスタの使用密度を増加させることにある。更に他の目的は、コンパイラの介入なしに

7

レジスタのセーブ及び復元を行うことにある。

【０００９】

【概要】本発明の１つの側面は、物理レジスタをスタティックレジスタとスタックレジスタに分割することにある。これにより、スタックレジスタがスタックをポインティングするベースレジスタまたはリロケーションレジスタを介して間接的にアドレスできるようになる。プロシージャ呼び出しの際にレジスタをセーブしプロシージャからの復帰の際にセーブされていたレジスタを復元するようにプロシージャに要請する代わりに、本方法では、全てのプロシージャが、その呼び出し元とは独立した一組のレジスタをスタックから割り当てる（プロシージャからの復帰の際にはその様なレジスタの割当てを解除してスタックへ戻す）ことができるようにする。もしそのような割り当てがスタックオーバーフローまたはアンダーフローをもたらさないなら、メモリアクセスは不要である。

【００１０】もしハードウエアが十分大きなスタックを実現するなら、呼び出されたプロシージャがローカルレジスタを直ちに入手できること、それがない場合にはレジスタのセーブ及び復元を必要とするメモリパイプが使えること、及びメモリとのトラフィックが減少することからもたらされるキャッシュ動作の改善により、システムスループット、資源利用及びプログラム実行時間の改善が期待される。

【００１１】プロシージャにより要求される（また、おそらくは必要な）レジスタの正確な個数がそのプロシージャに割り当てられる。もっと具体的には、本発明によれば、各プロシージャ及び各ループには正確にそれら特性に合う所要の個数のレジスタが割り当てられる。このようにして、レジスタは不必要に割り当てられることもなく不必要に保持／復帰されることもない。この特徴はレジスタの効果的な使用及び実行時間の短縮につながる。

【００１２】本発明はコンパイラの介入なしにディジタルコンピュータで動的にレジスタをプロシージャに割り当てる方法を含む。この方法は以下のステップを含む：複数のスタックレジスタを有する論理レジスタスタックを定義するステップ；ローカルリローケション項（local relocation term, “ｌｒｅｌ”と称する）を初期化して、前記論理レジスタスタックをコンピュータの前記物理レジスタセットにマッピングするためのオフセットを定義するステップ；第１のスタックポインタ値（ＴＯＬ）を初期化することにより、第１のプロシージャにより指定される任意の個数のスタックレジスタをローカルレジスタとして第１のプロシージャに割り当てて、前記論理レジスタスタック内の前記ローカルレジスタの範囲を区切るステップ；前記第１のプロシージャの実行の間の間のレジスタアクセス操作について、各ローカルレジスタを前記ローカルリロケーション項に応じて前記物理

8

レジスタセットにマッピングするステップ。

【００１３】第２のプロシージャを呼び出すための準備にあたっては、この方法は以下のステップを要する；第１のスタックポインタ値（ＴＯＬ）を“以前のＴＯＬ”（ＯＴＯＬ）と称される第２のスタックポインタ値としてストアするステップ；第１のスタックポインタ値（ＴＯＬ）をインクリメントすることにより、第１のプロシージャにより指定された個数のスタックレジスタを第１のプロシージャにパラメータ引渡レジスタとして追加して割り当てて、パラメータ引渡レジスタを含めるようにするステップ；呼び出されたプロシージャが参照するように、割り当てられたパラメータ引渡レジスタ内に選択されたパラメータをストアするステップ。我々はまた、パラメータ引渡レジスタをローカルリロケーション項に応じて物理レジスタセットにマッピングする。

【００１４】第２のプロシージャを呼び出すに当たって、この方法は更に第１のプロシージャパラメータ引渡レジスタを含む初期ローカルレジスタ空間を第２のプロシージャに割り当てるステップを含む。このステップにより、これらのレジスタにストアされたパラメータを、メモリを参照せずに第２のプロシージャから利用できるようにする。第２のプロシージャが必要とする個数の追加のスタックレジスタは、スタックポインタ値をインクリメントすることによりローカルレジスタとして第２のプロシージャに割り当てられる。この割り当ては、第１のプロシージャのローカルレジスタ内容を先ずメモリにセーブするという操作を経ずに実行される。第２のプロシージャからの復帰に当たっては、この方法では、スタックポインタ値をローカルレジスタの個数分デクリメントすることにより、ローカルレジスタの割り当てを解除する操作が行われる。このように、本発明には、ローカルレジスタ内容をセーブ及び復元することなしに第２のプロシージャを呼び出しまたそこから復帰することが含まれる。

【００１５】本発明の他の側面は、物理アドレスをレジスタファイルポートに提供するレジスタファイルポートアクセス回路である。この回路は仮想アドレスを受け取り、仮想アドレスをスタティックレジスタアドレス空間と比較してこの仮想アドレスがスタティックレジスタアドレス空間内にあるか否かを提示する。もしスタティックレジスタアドレス空間内にあれば、この回路はこの仮想アドレスを、対応レジスタにアクセスするための第１の物理アドレスとしてレジスタファイルポートに結合する。この回路には更に、受け取った仮想アドレスをローカルリロケーション項と結合して第２の物理アドレスを形成する回路と、仮想アドレスがスタティックレジスタアドレス空間内になければ第２の物理アドレスをアドレスとしてレジスタファイルポートに結合する手段も含まれる。アクセス回路は、ローカルリロケーション項を仮想アドレスと予め定められた物理レジスタの総数を法(m

odulo)とするモジューロ加算を行うように構成される。

【００１６】本発明の、上述した、またそれ以外の目的、特徴及び利点は、以下の図面を参照して進められる好適な実施例の詳細な説明からもっと容易に明らかになるであろう。

【００１７】

【実施例】図１はレジスタウインドウとして知られている、レジスタを割り当てる従来技術の方法を示す概念図である。以下において、参照番号は、図示されているアドレス空間モデルを参照するために使用される。図の参照番号をレジスタ番号と混同してはいけない。ここでは、物理レジスタ番号を示すために小文字のｒを使用し、論理または仮想レジスタスタック番号を示すために大文字のＲを使用する。省略記法“ＶＲ”は仮想レジスタを意味し、“ＰＲ”は物理レジスタを意味する。

【００１８】図１において、番号ｎ－１がついた第１のウインドウには、グローバルレジスタｒ０からｒ９及びローカルレジスタＲ１０からＲ３１が割り当てられる。新たなプロシージャが呼び出されると、別の一連のレジスタがこの新たなプロシージャに割り当てられる。番号１のウインドウを参照すると、レジスタｒ０からｒ９は、グローバルであるので、そのままである。呼び出しの後、６個のレジスタが前段のウインドウにオーバーラップして、呼び出し側のレジスタＲ１０からＲ１５までがレジスタＲ３１からＲ２６になる。１０個のレジスタはウインドウに含まれないので、各プロシージャからは一度に３２個のレジスタが見えていても、ウインドウ毎に１６個（３２－１０－６）の固有のレジスタがある。オーバーラップしたレジスタはパラメータ引渡のために使用される。同様に、ウインドウ番号ｎ＋１において、呼出側のレジスタのＲ１０からＲ１５（ウインドウｎ）は、呼び出し後、Ｒ３１からＲ２６となり、再び、６個の重複レジスタが提供される。従来技術で述べたように、固定サイズでの分割を行うレジスタウインドウ技術は、使用されないときでもセーブされるレジスタを生成する。

【００１９】本発明は、個々のプロシージャが物理レジスタのプールからあるいはそこへの任意の個数（論理レジスタスタックの数により制限される）のレジスタの割り当て及び割り当て解除を必要に応じて実行できるようにすることにより、プロシージャインターフェースにおいてセーブされまた復元されるレジスタの個数を最小限にする。このプールの中のレジスタは、以下に述べる間接あるいはリロケーションポインタを介してアクセスされる。レジスタスタックはレジスタで利用可能なソフトウエアスタックの頂部として見ることができ、従って、容易にかつ速やかにアクセスされる。そのようなレジスタの“動的割り当て(dynamic allocation)”は、コンパイラにより予め定められているのではなく、呼び出されたプロシージャ自身により制御される。

【００２０】本発明の一実施例では、１２８個の固定小数点及び１２８個の浮動小数点レジスタを設けてよい。典型的なハードウエアレジスタファイルは、６４個のスタティックレジスタ及び６４個の循環レジスタを有していてよい。レジスタファイルは独立した集積回路で実現してよいし、あるいはプロセッサデバイス上に実装してよい。物理レジスタファイル自体の実現の詳細は知られており、ここではあまり関係がない。本発明は固定小数点レジスタと浮動小数点レジスタの一方または両方に等しく適用可能である。説明の都合上、固定小数点レジスタに適用するものとして本発明を説明する。以下の説明では、次に示す用語を使用する。

【００２１】物理レジスタ（ＰＲ）：システムアーキテクチャから見える物理レジスタ。物理レジスタの実際の個数は単なる設計事項である。物理レジスタはレジスタファイル内で実現されると仮定している。

仮想レジスタ（ＶＲ）：命令中で指定されるレジスタ番号である。ＶＲ番号はＰＲ番号と同じであってよいし、あるいはＶＲを修正して、対応するＰＲのアドレスを決定してもよい（以下で説明する）。

スタティックレジスタ：グローバルレジスタ(Grobal Registers)とも称され、スタック操作あるいは循環に関与しないレジスタである。言い換えれば、これらレジスタのは、間接操作なしで、シラブル(syllable)で提供されるレジスタアドレスを使用して直接アクセスされる。以下の実施例では、ＶＲアドレス０から３１は、スタティックレジスタであるＰＲ０から３１にアクセスするために修正なしに使用される。

循環レジスタ(Rotating Registers)：これらのレジスタはソフトウエアパイプライン化に関与するための、プロシージャにより割り当てられたレジスタであり、循環レジスタベースＲＲＢ(Rotating Register Base)からのオフセットとしてアクセスされる。どのプロシージャも、いかなる時でも物理レジスタの個数によりほぼ制限される任意の個数の循環レジスタにアクセスできる。循環レジスタはリングとしてアドレスされる。

スタックレジスタ：スタック操作及び循環に関与するレジスタのプールである。スタックレジスタをアクセスするためのベースあるいは間接操作を指定するスタックポインタを使うことにより、スタックレジスタのプールはリングとして管理される。（従って、循環レジスタはリング内のリングとして管理される。）言い換えれば、もしＶＲ（ｉ）が物理的に実現されているスタックレジスタの中の最大のアドレスを有しているものに対応するならば、ＶＲ（ｉ＋１）は最小アドアドレスを有するスタックレジスタに対応する。以下に述べる実施例では、９６個のスタックレジスタ（Ｒ３２からＲ１２７）が存在する。循環レジスタはスタックレジスタのプールから抜き出される。ＶＲｓ３２から１２７は、ベースポインタにより修正されて、対応するＰＲを決定する。

ローカルレジスタ(Local Registers)：現プロシージャからアクセス可能なスタックレジスタである。

【００２２】上述の循環レジスタベース（ＲＲＢ）に加えて、好ましい実施例はスタックレジスタファイルの中をポインティングする次のような追加の間接つまりベースレジスタを維持する。（好ましくは、プロシージャは次のベースレジスタの各々の２つのコピーを有する。その１つは固定小数点スタックであり、他方は浮動小数点スタックである。）

有効レジスタ群の下限（底）ＢＯＶ：レジスタスタックを介してアクセス可能なソフトウエアスタックの深さをマーキングするスタックポインタである。ＢＯＶを越えて割り当て処理を実行するとスタックのオーバフローを招き、ＢＯＶを越えて割り当て解除処理を実行するとスタックのアンダフローを招く。以下で更に説明する。

ローカルレジスタ群の下限ＢＯＬ：現プロシージャからアクセス可能なスタックレジスタ群の一方の端の境界を示すスタックマーカである。現プロシージャの全てのスタックレジスタはＢＯＬに関してアクセスされる。一般に、Ｐがスタックレジスタの総数であり、ｉ及びｊが、ｉ＜３２の場合ｊ＝ｉ、ｉ≧３２の場合ｊ＝［（ＢＯＬ＋ｉ－３２）ｍｏｄ Ｐ］＋３２によって関連付けられるとするとき、ＶＲｉはＰＲｊにアクセスする（ここで、ＰＲ０からＰＲ３１はスタティックレジスタであると仮定している）。好ましい実施例において、ＢＯＬはデフォルトで最初のスタックレジスタ（ＰＲ３２）を指す。

ローカルレジスタ群の上限（頂上）ＴＯＬ：現プロシージャからアクセス可能なレジスタ群の他端の境界を示すスタックマーカである。プロシージャがローカルエリア内に存在しないレジスタ、すなわちＢＯＬ境界からＴＯＬ境界までの範囲の外にあるレジスタにアクセスしようとすると例外を引き起こす。

ローカルレジスタ群の以前の上限ＯＴＯＬ：パラメータレジスタの割り当ての行われる以前のＴＯＬの値である。ＴＯＬとＯＴＯＬの間の一連のレジスタは割り当てられたパラメータレジスタである。

循環レジスタ群の下限ＢＯＲ：循環にかかわるスタックレジスタ群の一端の境界を示すスタックマーカである。

循環レジスタ群の上限ＴＯＲ：循環にかかわるスタックレジスタ群の他端の境界を示すスタックマーカである。

【００２３】一般に、レジスタスタックにおける利益はスタックレジスタにだけあてはまり、コンパイラは依然としてスタティックレジスタのための呼び出し元／呼び出し先、セーブ／復元戦略を採り入れ続けねばならない。全ての議論及び説明は固定小数点スタックと浮動小数点スタックに等しく適用される。各スタックは自分自身のベースレジスタセットを有する。固定小数点レジスタファイル及び浮動小数点レジスタファイルは各々別々に制御される。本発明を説明するために、固定小数点スタックを詳細に述べる。

【００２４】ローカル、パラメータ及び循環レジスタは新たに定義された操作であるａｌｌｏｃを実行することにより割り当てられまたは割り当て解除される。ローカルレジスタが割り当てられ／割り当て解除される場合には、レジスタの割り当て及び割り当て解除により、ＴＯＬ及びＯＴＯＬが修正される。ＴＯＬ及びＯＴＯＬは割り当て及び割り当て解除されるローカルレジスタの個数だけインクリメント／デクリメントされる。パラメータレジスタの割り当て／割り当て解除により、ＴＯＬが修正される。ＴＯＬは、以下の図３Ａ－Ｉに示されるように、割り当て及び割り当て解除されるパラメータレジスタの個数だけインクリメント／デクリメントされる。レジスタの割り当て及び割り当て解除は、スタックがオーバフロー／アンダフローするときＢＯＶにも影響を及ぼすかもしれない。

【００２５】循環レジスタの割り当て／割り当て解除により、ＢＯＲ、ＴＯＲ、ＴＯＬ、及びＯＴＯＬが修正される。循環レジスタが割り当てられる際、ＢＯＲはＴＯＬの内容にセットされる。ＴＯＬ、ＯＴＯＬ及びＴＯＲはＴＯＬと割り当てられる循環レジスタの個数の合計にセットされる。循環レジスタの割り当て解除より、逆方向の修正が行われる。上述した機構により、プログラムに影響を与えることなしに、例えば装置モデル毎に物理レジスタの個数を変えることができる。

【００２６】プロシージャ呼出し（すなわち、ブランチアンドリンクの実行）の実行に当たって、各種のベースレジスタの現在の状態がパラメータレジスタ０にストアされる。よってコンパイラは呼び出されたプロシージャへ渡す／呼び出されたプロシージャから送られるパラメータの個数以外に、追加のパラメータレジスタを１個割り当てなけらばならない。更にＢＯＬはＯＴＯＬの値にセットされ、ＯＴＯＬはＴＯＬの値にセットされる。プロシージャからの復帰に当たっては、各種のベースレジスタはパラメータレジスタ０にストアされた値に設定しなおされる。プロシージャからの復帰は、呼び出されたプロシージャのローカルレジスタ及び循環レジスタの割り当て解除を伴うので、スタックアンダーフローを招くことがある。

【００２７】割り当てを行っている際にＴＯＬがＢＯＶを越えようとするとき、スタックはオーバーフローしたと言われる。全ての演算が、スタックに実装されているレジスタの個数を法として行われるということを想起されたい。あるいは、モジューロ加算(modulo-plus)機能を使用して固定小数点レジスタアドレス空間を飛び越してもよい。同様に、割り当て解除を行っている際にＢＯＬがＢＯＶを越えようとするとき、スタックはアンダーフローした言われる。オーバーフロー／アンダーフローの発生はハードウエアで検出され、トラップハンドラが適切にスタックレジスタをソフトウエアスタックに追出

すために／ソフトウエアスタックからスタックレジスタへの復元を行うために呼び出される。

【００２８】上述した機構により、スタックオーバーフロー及びアンダーフローを予期してレジスタスタックの内容の流し出しまたそこへの充填をバックグラウンドで行うハードウエア（ソフトウエア）を使用することができる。従って、従来の意味でのスタックオーバーフロー及びアンダーフローは、後述するところの我々がレジスタ清浄化(register cleaning)と呼んでいる処理により回避することができる。

【００２９】割り当て処理操作は、例を使うことによって最も良く説明できる。プロシージャＡでＢＯＬが物理レジスタ３８をポインティングし、ＴＯＬが物理レジスタ４７をポインティングすると仮定する。するとプロシージャＡは１０個のローカルレジスタを有する。プロシージャＢの呼び出しに先立って、プロシージャＡは４個のパラメータレジスタを割り当てる。これにより、ＯＴＯＬが４７にセットされ、ＴＯＬは５１にセットされる。ブランチアンドリンクが実行されるとき、これらのベースレジスタの値がパラメータレジスタ０、すなわち物理レジスタ４８にパックされる（以下の制御レジスタＡについての説明を参照）。更に、ＢＯＬは４７にセットされる。呼び出されたプロシージャ（Ｂ）のローカル空間の下限をＯＴＯＬに配置することにより、パラメータレジスタは両者に共通となり、またこれらのレジスタ部分はＢのローカル領域の下限部となる。ＯＴＯＬ及びＴＯＬは５１にセットされる。プロシージャＢに１０個のローカルレジスタを割り当てると仮定する。この場合、ＴＯＬ及びＯＴＯＬを６０にセットする。プロシージャＢから復帰すると、ＢＯＬ、ＴＯＬ及びＯＴＯＬは夫々初期値３８，４７，５１にリセットされる。

【００３０】ある実施例では、プロシージャＢによりプロシージャＡへ戻される値を物理レジスタ４８から５１に入れることができる。その代わりに、復帰値をスタティックレジスタに入れることもできる。これにより、復帰後直ちにパラメータレジスタの割り当てを解除して次のプロシージャ呼び出しのために利用できる。

【００３１】プロシージャが呼び出されるとき１つのパラメータレジスタ（ここでは４８）を使ってポインタ値がストアされるということにも注意されたい。もっと具体的に言えば、復帰情報は制御レジスタに記憶され（以下で更に説明する）、コンパイラは、復帰前に、復帰情報をローカルレジスタ領域にコピーし、また制御レジスタにストアすることが必要とされる。好ましくは、ＴＯＬ及びＯＴＯＬ値自体はセーブされないが、それらの値を計算できるようにする代わりの値、公称上、以前の値に対するオフセット、がセーブされる。オフセット値を使用することによって、これらの値を任意のレジスタに記憶できるようになる。かくして、レジスタを任意量だけ循環しても、上記機構は依然として正確に動作する。

他の実施例では、この目的のためにパラメータレジスタを余分に割当てて、パラメータ引渡に使用できるレジスタの総数を呼び出し側のプロシージャにより割り当てられた実際の数に等しくなるようにする。循環レジスタの割り当て／割り当て解除は、同様の態様で動作する。

【００３２】各レジスタスタックは、スタックオーバーフローの際にはレジスタがセーブされ、またスタックアンダーフローの際にはレジスタが読み出される固有のソフトウエアスタックを有する。従って、各レジスタスタックはまさに適当なソフトウエアの先端部分を表す。

【００３３】上述の方法を実現するに当たって、以下の制御レジスタを提供するのが好都合である：

制御レジスタＡ：これは固定小数点スタック用の各種のベースポインターＢＯＶ，ＢＯＬ，ＴＯＬ，ＢＯＲ，ＴＯＲ及びＯＴＯＬーを収容する。

制御レジスタＢ：これは浮動小数点用の別のベースポインターＢＯＶ，ＢＯＬ，ＴＯＬ，ＢＯＲ，ＴＯＲ及びＯＴＯＬーを収容する。

制御レジスタＣ：これは固定小数点レジスタのためのレジスタスタックをバックアップしているソフトウエアスタックのメモリアドレスを含む。

制御レジスタＤ：これは浮動小数点レジスタのためのレジスタスタックをバックアップしているソフトウエアスタックのメモリアドレスを含む。

プロシージャ呼び出しの実行の準備として、適当なベースポインタがパラメータレジスタ０にストアされることを想起されたい。

【００３４】図２に目を向けると、Ｒ０からＲ１２７までの番号が付与されたレジスタセットについての論理アドレス空間モデルが示されている。スタティックレジスタ５０（Ｒ０からＲ３１）は例えばグローバル値のためにリザーブされ、ローカルレジスタ割り当て機構には関与しない。スタックレジスタＲ３２からＲ１２７は参照番号５８により示される（この参照番号はアドレスではないことに注意）。このモデルにより示される仮想アドレスはソフトウエアプロシージャから見たレジスタスタックを示す。仮想アドレス（ＶＲ）は、以下に述べられる物理レジスタファイルにアクセスするために実際のつまり物理レジスタアドレス（ＰＲ）に変換される。初期状態では、割り当てられていないアドレス空間６０がレジスタスタック全体を構成している。

【００３５】図３Ａから図３Ｉは、呼び出された一連のプロシージャから見える仮想アドレス空間を示している。呼び出されたプロシージャは、各図の上部に書かれたＡ、Ｂ、Ｃ、Ｄで示している。このモデルで方向「上」、「下」、及び表記「上限」、「下限」は任意である。例えば、ローカルレジスタを「上限」、ここではＲ１２７から下向きに割り当て、スタックの下限（ＶＲ３２）に到達したとき回り込みを行うこともできる。我々は、Ｒ３２から上向きに割り当てることによって本発

15

明を説明する方を選択した。動作の原理は、一貫性を保つ限り同一である。

【００３６】図３Ａに目を向けると、第１のプロシージャＡへの呼び出しを行った後の図２の論理アドレス空間が示されている。論理アドレス空間（すなわち一連の仮想レジスタ）６２はプロシージャＡに対してローカルなものとしてプロシージャＡに割り当てられている。ＢＯＬ(Bottom of Local)ポインタはプロシージャＡのローカル空間の下限を示し、ＴＯＬ(Top of Local)はプロシージャＡのローカルアドレス空間の上限を区切っている。ＢＯＶ(Bottom of Valid)は、ＢＯＬの値に初期化され、現在割り当てられている空間の範囲を区切っている。参照番号６０は未だ割り当てられていない仮想レジスタ（つまりアドレス空間）、すなわちＴＯＬよりも上の空間またはＢＯＶよりも下の空間を示す。図３Ｂにおいて、プロシージャＡは次に呼び出されるプロシージャへのパラメータ引渡のためにパラメータ空間６４を割り当てる。パラメータ空間６４はプロシージャＡに割り当てられているローカルアドレス空間をインクリメントする。この様子は、それに合わせてＴＯＬポインタがパラメータ空間６４の上限に上向きに調整されていることに示されている。ポインタＯＴＯＬはパラメータレジスタの割り当ての前のＴＯＬ値を示している。

【００３７】プロシージャＡは次にプロシージャＢを呼び出す。注意したように、ポインタ（制御レジスタＡ）は第１のパラメータレジスタにストアされる。図３Ｃを参照すると、いつものように、プロシージャＢは、ＢＯＬポインタで表わされるように、スタックの下限（ＶＲ３３）で始まるローカル（仮想）アドレス空間を割り当てる。プロシージャＢのローカル空間の最初の部分は、パラメータ空間６４がプロシージャＡとＢに共通になるように、プロシージャＡのパラメータ引渡レジスタ６４にマッピングされる。呼び出されたプロシージャのローカル空間は常にスタックの下限（ＢＯＬ）で始まり、この空間はまた呼び出し側のプロシージャのパラメータ空間で始まる。

【００３８】従って、プロシージャ呼び出しは、パラメータ引渡空間（例えば６４）が下限側に向かうように仮想レジスタスタックを押し込む、つまりプッシュダウンすると考えることができる。呼び出し側プロシージャのパラメータ空間（例えば６２）は、図３Ｃのアドレス空間の上限部分に回り込み、ＢＯＶ(Bottom of Valid)ポインタを調整することにより区切られている。プロシージャＢも（純粋にローカルな）レジスタ６６を更に割り当て、ＴＯＬポインタにより区切っている。上と同様に、残りの割り当てられずに残っているアドレス空間は６０によって示される。

【００３９】図４Ａから図４Ｉは、レジスタファイルのような物理アドレス空間をモデル化している。この段階で、図３Ａから図３Ｉにモデル化された仮想アドレス空間と物理アドレス空間の関係を質的に考慮することは役に立つ。図４Ａを参照すると、ＢＯＬ及びＢＯＶポインタはレジスタファイルアドス空間の原点を示している。これらは例えば物理アドレス０で良い。プロシージャＡの仮想アドレス空間６２（図３Ａ）はＴＯＬポインタで区切られている物理アドレス空間１０２（図４Ａ）に対応する。図４Ｂはまた、プロシージャＡにより割り当てられ、図３Ｂの仮想アドレス空間６４に対応するパラメータ空間１０４も示している。参照番号１００は物理アドレス空間モデル中でに現在割り当てられていないアドレス空間を示している。一般に、図３Ａから図３Ｉの参照番号に４０を加算すれば、夫々図４Ａから図４Ｉの対応する参照番号になる。

16

【００４０】図４Ｃは、ＴＯＬポインタを調整することにより図３Ｃのプロシージャ Ｂのローカルアドレス空間６６に対応するアドレス空間１０６が更に割り当てられたことを示している。従って、呼び出されたプロシージャの仮想アドレス空間は常にレジスタスタックの下限から始まるが、物理レジスタファイル内ではこれに対応するデータのリロケーションはないことが観察できる。リロケーションを実際に行う代わりに、図４Ａから図４Ｉに示されるように、追加されるレジスタは、前に割り当てられた物理アドレス空間に影響を与えることなしに呼び出されたプロシージャにより必要に応じて割り当てられる。次に、これ以上のプロシージャ呼び出しを考慮するために図３Ｄを参照する。

【００４１】プロシージャＢは図３Ｄに示されるように、ＴＯＬポインタを調整することによりパラメータ引渡アドレス空間７０を割り当てる。プロシージャＡと共通のパラメータアドレス空間６４を有する残りのローカルアドレス空間６６は影響されない。図３Ｄに示され、ＢＯＶアドレスポインタによって表わされるプロシージャＡのローカルアドレス空間６２は、アドレス空間の上限部にそのまま残っている。

【００４２】図３Ｅを参照すると、プロシージャＢは更に別のプロシージャＣを呼び出す。プロシージャＣについての論理アドレス空間は以下の部分で構成される：論理アドレス空間の下限（ＢＯＬ）で始まり、アドレス空間７０はプロシージャＢと共通のパラメータ引渡空間である。プロシージャＣはＴＯＬポインタで表わされるローカルアドレス空間７２を割り当てる。呼び出し側プロシージャ（Ｂ）のローカルアドレス空間６４，６６は、プッシュダウンされ、図３Ｅのモデルの上限部分に回り込んむ。プロシージャＡローカル空間６２は現在の呼び出しに適応するためにプッシュダウンされ、またＢＯＶがそれに従って動かされる。言い換えれば、レジスタスタックは論理的に循環する。いつものように、残りの割り当てられていないアドレス空間は６０で示される。

【００４３】図４Ｄ及び図４Ｅは夫々図３Ｄ及び図３Ｅでモデル化された論理アドレス空間に対応する物理アド

レス空間を示している。図４Ｄを参照すると、パラメータ引渡空間１１０は、図３Ｄでプロシージャ Ｂにより割り当てられた仮想パラメータ空間７０に対応する。同様に、図４Ｅのローカルアドレス空間１１２は、図３Ｅでプロシージャ Ｃにより割り当てられた仮想ローカルアドレス空間７２に対応する。

【００４４】ここで図３Ｆを参照すると、プロシージャ Ｃは前に割り当てられたローカル空間７２に加えて循環レジスタアドレス空間７４を割り当てる。ＴＯＬ（及びＴＯＲ；図３Ｇ参照）ポインタは循環レジスタ空間の上限を示し、ＢＯＲは循環レジスタ空間の下限を示している。仮想アドレス空間６２，６４及び６６は影響されない。図４Ｆは、ＢＯＲ及びＴＯＬで境界付けられる物理アドレス空間１１４の対応する割り当てを示している。循環レジスタの個数は、ソフトウエアのパイプライン化されたループの特性に応じて変えられる。プロシージャが現在利用可能なアドレス空間を越える循環レジスタ空間を割り当てようとする場合のみ、レジスタオーバーフローが起こる。この場合については、以下に述べられる。

【００４５】次に、図３Ｇを参照すると、他のプロシージャの呼び出しを見越してプロシージャ Ｃは循環レジスタ空間７４の上限の上にパラメータ引渡空間７６を割り当てる。パラメータ空間を区切るためにＴＯＬを調整する。論理アドレス空間６０はまだ割り当てられていないレジスタである。図４Ｇは、プロシージャ Ｃが他のプロシージャにパラメータを通すための物理アドレス空間１１６の対応する割り当てを示している。

【００４６】図３Ｈはもう１つのプロシージャ Ｄがプロシージャ Ｃにより呼び出された後の仮想アドレス空間モデルを示している。前にプロシージャ Ｃにより割り当てられたパラメータ空間７６は、いつものように、プロシージャ Ｄのローカルアドレス空間の下限に現れている。更に、プロシージャ ＤはＴＯＬポインタを調整することによりローカルアドレス空間７８を割り当てる。呼び出しプロシージャ、すなわちプロシージャ Ｃ、についてローカルなアドレス空間（共通パラメータ引渡空間７６を除く）は、図３Ｈの７０，７２及び７４で示されるように、プッシュダウンされ、モデルの上限部に回り込んでいる。プロシージャ Ｂについてローカルなアドレス空間６６及び６４はこれに従ってプッシュダウンされる。同様に、プロシージャ Ａのローカル空間６２は、スタック上で更にプッシュダウンされ、未だ以って割り当てられていないアドレス空間６０を残して、ＢＯＶ(Bottom of Valid)ポインタにより区切られる。図４Ｈは、プロシージャ Ｄがローカルレジスタとして使用する物理アドレス空間１１８の対応する割り当てを示す。

【００４７】プロシージャ Ｄは次に、図３Ｈの６０により示される利用可能なアドレス空間を越えて循環レジスタ空間を割り当てようとする。これはレジスタオーバー

フロー条件を引き起こす。その結果、ＢＯＶポインタの上方にあるメモリ空間の部分がメモリ（図示せず）にセーブされる。セーブされる部分は論理アドレス空間６２と６４及び６６の一部分を含む。ＢＯＶポインタはオーバーフローセーブ操作の結果繰り上がり、これにより空いている空間を追加する。この結果の割り当てられていないアドレス空間６０は、プロシージャ Ｄの循環レジスタの要求に合致するよりも大きい。この結果は図３Ｉに示されており、ここで８０はプロシージャ Ｄの循環レジスタ空間を示している。

【００４８】図３Ｉを参照すると、プロシージャ ＤはＢＯＲ及びＴＯＬポインタで区切られている循環レジスタアドレス空間８０を割り当てし終わっている。この場合、必要最小限の空間より幾分大きいものがメモリにセーブされている。その結果、割り当てられていない部分６０が残っている。これは単に最小限の今すぐに必要というのではなく、予め定められた個数のアドレスを動かすようにオーバーフローセーブ機構を構成していることから起こる。上述のセーブ操作によってリロケーションされたアドレスの個数は、好ましくは従属ハードウエアで効率よく実現できるように選択される。この結果現れるヒステリシスは、使用時に必要になるメモリ参照の回数を減少させ得る。他の実施例では、保留されている割り当てを調整するために十分なアドレス空間のみをセーブする。レジスタオーバーフロー及びセーブ後であって、所要の循環レジスタを割り当てた後の物理アドレスモデルは図４Ｉに示される。ここで１２０はプロシージャ Ｄの循環レジスタ空間を示している。

【００４９】レジスタオーバーフローセーブ及び復元機構の詳細は知られている。しかし、本発明の他の側面は、完全にレジスタオーバーフローを回避するための、仮想アドレスレジスタシステムとともに作動する「清浄化(cleaning)」機構にある。「清浄な(clean)」レジスタは、メモリ内容の正確なコピーを有するレジスタとして定義される。逆に、「汚染した(dirty)」レジスタはメモリ内容の信頼できるコピーを持っていない。汚染したレジスタは恐らくは有効なものである。すなわち、おそらく現在割り当てられている。清浄なレジスタ空間はＢＯＣ(Bottom ofClean)及びＴＯＣ(Top of Clean)ポインタで表わされている。ＢＯＣは本質的にはＢＯＶと同じである。初期状態では、定義により、レジスタ内容がメモリにコピーされるまでは清浄なレジスタが存在しないので、ＴＯＣはＢＯＣに等しい。レジスタの清浄化はバックグラウンドにおいてトランスペアレントに、すなわちもしそうでなければ遊んでしまうプロセッササイクルを「盗む」ことにより実行される。

【００５０】ＴＯＣがＢＯＬより小さいとき、あるレジスタはメモリをまだ更新していない。レジスタ清浄化機構は次のレジスタ、すなわちＴＯＣ＋１での値をメモリにコピーする。この機構は次にＴＯＣをインクリメント

し、その結果、ＴＯＣは常に先頭の清浄なレジスタをポインティングする。一般にローカルレジスタは頻繁に汚染されるので、無視することができる。そこでＢＯＬまでしか清浄化しないことが好ましい。清浄化操作はソフトウエアからはトランスペアレントであり、ここで説明するレジスタ割り当て／割り当て解除方法及び装置とは独立である。

【００５１】レジスタファイルポートアクセス回路

ケース１：スタティックレジスタアクセス

図５は本発明に関するレジスタファイルポートアクセス回路１４０のブロック図を示している。物理レジスタ、例えば１２８個のレジスタは、レジスタファイル１４４のような一連のハードウエアレジスタファイル中に提供される。図５に示されるタイプのアクセス回路が各レジスタファイルポートに対して提供される。回路機能の１つは、例えばソフトウエアプロシージャにより提供されるアドレスである論理アドレスＲを、レジスタファイルにアクセスするための対応する物理レジスタアドレスにマッピングすることである。回路１４０において、論理レジスタアドレスＲがライン１４２上に入力され、マルチプレクサ１４６への３つの入力の１つに結合される。

【００５２】コンパレータ１５０は、Ｒの値を具体的なアプリケーション中でのグローバルつまりスタティックレジスタの個数（この例では３２）に等しい定数と比較し、この論理アドレスがスタティックレジスタの範囲の中に入るかどうか（つまり、Ｒ＜３２かどうか）を判定する。もしＲが３２より小さいときは、提示されたアドレスがスタティックレジスタの範囲内にあり、コンパレータ１５０からの出力はマルチプレクサ制御ライン１５２をアサートして、ＭＵＸ１４６が物理アドレスとしてレジスタファイルに入力するものとして値Ｒ自体を選択するようにする。言い換えれば、Ｒはスタティックレジスタについては修正されない。上述のように、スタティックレジスタはスタックレジスタ操作には関与しない。

【００５３】ケース２：レジスタスタックアクセス

もしＲが３２以上（かつｑｕｐまたはｐｄｎがアサートされていない）ならば、Ｒは有効なレジスタスタック仮想アドレスであり、物理レジスタアドレスにマッピングされなければならない。さしあたり、循環レジスタが現プロシージャに割り当てられていないと仮定する。この場合、モジューロ加算を行う加算器(modulo-plus adder)１５４の出力はＭＵＸ１４６を介してレジスタファイル１４４に供給される物理アドレスとして選択される。モジューロ加算を行う加算器１５４は、物理レジスタアドレスを決定するためにモジューロ演算を使用して、論理アドレスＲをローカルリロケーション項（“ｌｒｅｌ”）、つまりオフセットと結合する。リロケーションの加算は、スタック内に物理的に実装されているレジスタ数についての法を取って実行される。ローカルリロケーション項ｌｒｅｌは、（ＢＯＬ値）－（固定されたレジスタの個数）に等しい。ｌｒｅｌは任意であり、予め定められたリロケーションオフセット量またはブロックサイズに限定されない、ということに注意されたい。このように、所与のプロシージャにより割り当てられた個数丁度のレジスタが使用される。逆に、復帰に当たっては、丁度同じ個数のレジスタが割り当て解除される。

【００５４】説明のため、ハードウエアレジスタの総数が１２８個で、レジスタ０から３１が固定されたレジスタであり、レジスタスタックが９６個のレジスタを備えていると仮定する。次に、仮想スタックアドレスＲ＝４４及びＢＯＬ＝４０であると仮定する。すると、

Ｒと（ＢＯＬ－３２）をモジューロ加算したもの＝（４４＋８）ｍｏｄ ９６＝５２

である。この例では、モデューロ加算による回り込み（ラップアラウンド）は起こらない。しかし、もしＢＯＬ＝９０なら、

Ｒと（ＢＯＬ－３２）のモジューロ加算＝（４４＋５８（つまり１０２））ｍｏｄ ９６＝６

である。ただし、このモジューロ加算演算の結果は更にＲ０からＲ３１を飛び越えるようにさせるので、最終的には３８となる。一般には、Ｐをスタックレジスタの総数とし、ｉとｊが、ｉ＜３２ならｊ＝ｉ、ｉ≧３２ならｊ＝［（ＢＯＬ＋ｉ－３２）ｍｏｄ Ｐ］＋３２という関係にあるとき、ＶＲｉはＰＲｊをアクセスする。

【００５５】ケース３：レジスタ復元及び清浄化

レジスタファイルポートアクセス回路１４０はまた、レジスタ復元及びレジスタ「清浄化」(register cleaning)のためにアクセスができるようにする。制御信号ｑｕｐは、すでに上書きされているがレジスタファイル内で再度有効(valid)にしなければならないレジスタを復元するためにメインメモリから読み出すことを指示する。この制御信号は、もっと多くの有効なレジスタを提供するためにスタックアンダーフローにともなって使用される。ｑｕｐがアサートされると、この信号はＭＵＸ１４６を制御して、レジスタファイルをアクセスするためのアドレスとしてＱＵＰを選択する。ＱＵＰは復元されるべき次のレジスタのアドレス、すなわちＢＯＶ－１である。

【００５６】ＱＵＰはローカル空間の外側を清浄にするための次のレジスタのアドレスである。これは次の使用可能な、つまり有効ではないレジスタである。よって、ＱＵＰアドレスは単にＴＯＬ＋１である。メインメモリの内容はレジスタファイルのこのアドレスにコピーされ、定義によりそのレジスタを清浄にする。ＴＯＣは常に清浄な空間の上限をポインティングするようにインクリメントされる。

【００５７】制御信号ｑｄｎはメインメモリにその内容をコピー（書き込み）することによりレジスタを清浄にすることを指示する。ｑｄｎがアサートされると、それはＱＤＮをレジスタファイルをアクセスするためのアド

レスとして選択するようにＭＵＸを制御する。ＱＤＮは清浄にすべき次のレジスタのアドレス、すなわちＴＯＣ＋１である。レジスタ清浄化機構はこのレジスタの内容をメインメモリにコピーする。この機構は次にＴＯＣをインクリメントし、ＴＯＣが清浄なレジスタ群のうちの上限にあるものを常にポインティングするようにする。

【００５８】ＱＵＰ及びＱＤＮは互いに排他的である。ポートはこれらの中の一方または他方の何れかを備えるが、両方ということはない。ＱＵＰはレジスタファイル内のストアポートで実現され、常にメモリからの読み出しを行う。ＱＤＮはレジスタファイル内の読み出しポートで実現され、常にメモリへの書き込みを行う。図５及び図６中の“ＱＵＰまたはＱＤＮ”という表記法は、図面を増やさずにこの相互の排他性を伝えようとするものである。

【００５９】循環レジスタ実装付きのレジスタファイルポートアクセス

図６において、レジスタファイルポートアクセス回路１６０がブロックの形態で示されている。図６の回路１６０は図５の回路１４０と共通のいくつかの素子を備えている。ここで同じ様な参照番号は共通の回路素子を示す。共通の特徴の記載は省略する。図６はレジスタスタック内で循環レジスタを実現するための追加の回路素子を含んでいる。以前と同様に、論理アドレスＲは入力ノード１４２に提供される。コンパレータ１６４は論理アドレスＲとＢＯＲ(Bottom of Rotating)ポインタを比較する。別のコンパレータ１６６はＲとＴＯＲポインタを比較する。ＲがＢＯＲより大きく、ＴＯＲより小さいときは、論理アドレスは循環レジスタとして現プロシージャに割り当てられたレジスタを指している。

【００６０】物理アドレスｒはＲに循環リロケーション項(rotating relocation term)ｒｒｅｌを加算したものに等しい。循環リロケーション項ｒｒｅｌは、循環レジスタ内で回り込みが起きないと仮定すると、循環レジスタセット内の循環を考慮に入れれば、ローカルリロケーション項ｌｒｅｌに循環レジスタベース値（ＲＲＢ）を加算したものに等しい。従って、

ｒ＝Ｒ＋ｌｒｅｌ＋ＲＲＢ

となる。しかし、循環レジスタ内で回り込みが起これば、

ｒ＝Ｒ＋ｌｒｅｌ＋ＲＲＢ－（ＴＯＲ－ＢＯＲ）

となる。

【００６１】ここでＴＯＲ－ＢＯＲは循環レジスタセットの大きさを与える。アクセス回路１６０で、Ｒは加算器１７０でｒｒｅｌに加算され（前述のモデューロ加算演算を使用）、その結果はＭＵＸ１６２に与えられる。ｌｒｅｌ及びＲＲＢはＲよりも前にわかるので、リロケーション項ｒｒｅｌを前以って計算しておくことができる。回り込みが起こる場合、別のリロケーション項ｒｒｅｌ＃がモジューロ加算器１７２でＲに加算され、その

結果がＭＵＸ１６２に与えられる。ここで、ｒｒｅｌ＃はｌｒｅｌ＋ＲＲＢ－（ＴＯＲ－ＢＯＲ）に等しい。ｒとしてＭＵＸ１６２で選択される値は、以下のように決定される。何も手を打たないと論理アドレスが循環レジスタの境界を越えてしまう場合に循環レジスタ内で回り込みが起こる。従って、問題は、

Ｒ－ＢＯＲ＋ＲＲＢ＞ＴＯＲ－ＢＯＲ

かどうか？ ということである。代数学から、このテストはＲ＞ＴＯＲ－ＲＲＢかどうか？ というテストと等価である。これは、ＲとＴＯＲ－ＲＲＢとを比較する、図６のコンパレータ１６８により判定される。その判定結果が真であれば、コンパレータ１６８は物理アドレスｒとしてモデューロプラス加算器１７２の出力を選択するようにＭＵＸ１６２を制御する。結果が偽であれば、循環レジスタは回り込まなかったのであり、コンパレータ１６８は物理アドレスｒとしてモデューロプラス加算器１７０の出力を選択するようにＭＵＸ１６２を制御する。

【００６２】種々の回路は回路１４０または１６０の所要機能を達成するために工夫される。例えば、清浄なアドレスという特徴はあるアプリケーションでは実現されるが他のアプリケーションでは実現されない。あるアプリケーションは従属(subject)スタック内の循環レジスタを提供しないが、このようなアプリケーションでは図５のような回路で十分である。他のアプリケーションでは、循環オフセットＲＲＢを他の場所で計算し、必要に応じて結果を加算器１７０に提供してよい。夫々の実現形態の、性能上のトレードオフを持ちやすい具体的内容は、本明細書を読んだ当業者には自明であろう。レジスタファイルポートアドレッシングがクリティカルパスであるアプリーケーションでは、例えば高速の並列ハードウエアが提案される。

【００６３】図７は本発明を実現するためのレジスタファイルシステムの一例を概略的に示すブロック図である。ｑｕｐ／ｐｄｎ，ＴＯＲ，ＢＯＲ，ＲＲＢ，ＱＵＰ／ＱＤＮ，ｌｒｅｌ，ｒｒｅｌ及びｒｒｅｌ＃が付された一連のレジスタは、対応するポインタ値を維持するために提供される。これらのレジスタは、所要のポインタ値を、上述したレジスタファイルポートアクセス回路１６０のような再マッピング回路に供給するためにバス１７６を介して結合されている。レジスタファイル１４４内で使用される各レジスタファイルポート毎に、このような再マッピング回路が１つ提供される。この一般的な構成の種々の変形は、前述の目的及び動作を見れば、熟達したハードウエア設計者には明らかであろう。例えば、複数のポインタ値をもっと少ないレジスタ内に収めてもよい。選択された途中の値または途中のアドレスを、性能を最適化するために前以って計算しておいてもよい。種種の作業のハードウエアとソフトウエア（マイクロコードを含む）への振り分けのような他の変形は、

特定の実現形態についての設計上のトレードオフと適応の問題であり、これらは皆上記実施例と等価であると考えられる。

【００６４】本発明の好ましい実施例で発明の原理を図示、説明したが、本発明はその原理から逸脱することなく配置及び細部を修正できることは当業者には明らかである。本願特許請求の範囲の精神及び範囲内で得られる全ての修正を本願の技術的範囲である。

【００６５】

【効果】以上詳細に説明したように、本発明によればレジスタを有効に活用でき、プロシージャ呼出し／復帰に伴うレジスタセーブ／復元のためのメモリアクセスを最小化することができる方法及び装置が提供される。

【図面の簡単な説明】

【図１】レジスタウインドウを示す概念図。

【図２】一連のレジスタの論理アドレス空間モデルを示す図。

【図３Ａ】本発明の操作を示す図２のレジスタセットの論理アドレス空間モデルを示す図。

【図３Ｂ】本発明の操作を示す図２のレジスタセットの論理アドレス空間モデルを示す図。

【図３Ｃ】本発明の操作を示す図２のレジスタセットの論理アドレス空間モデルを示す図。

【図３Ｄ】本発明の操作を示す図２のレジスタセットの論理アドレス空間モデルを示す図。

【図３Ｅ】本発明の操作を示す図２のレジスタセットの論理アドレス空間モデルを示す図。

【図３Ｆ】本発明の操作を示す図２のレジスタセットの論理アドレス空間モデルを示す図。

【図３Ｇ】本発明の操作を示す図２のレジスタセットの論理アドレス空間モデルを示す図。

【図３Ｈ】本発明の操作を示す図２のレジスタセットの論理アドレス空間モデルを示す図。

【図３Ｉ】本発明の操作を示す図２のレジスタセットの論理アドレス空間モデルを示す図。

【図４Ａ】図３Ａの論理アドレス空間モデルに対応する物理アドレス空間モデルを示す図。

【図４Ｂ】図３Ｂの論理アドレス空間モデルに対応する物理アドレス空間モデルを示す図。

【図４Ｃ】図３Ｃの論理アドレス空間モデルに対応する物理アドレス空間モデルを示す図。

【図４Ｄ】図３Ｄの論理アドレス空間モデルに対応する物理アドレス空間モデルを示す図。

【図４Ｅ】図３Ｅの論理アドレス空間モデルに対応する物理アドレス空間モデルを示す図。

【図４Ｆ】図３Ｆの論理アドレス空間モデルに対応する物理アドレス空間モデルを示す図。

【図４Ｇ】図３Ｇの論理アドレス空間モデルに対応する物理アドレス空間モデルを示す図。

【図４Ｈ】図３Ｈの論理アドレス空間モデルに対応する物理アドレス空間モデルを示す図。

【図４Ｉ】図３Ｉの論理アドレス空間モデルに対応する物理アドレス空間モデルを示す図。

【図５】本発明の一実施例を実現するレジスタファイルポートアクセス回路を示すハードウエアブロック図。

【図６】本発明の、レジスタスタック内に循環レジスタを有する他の実施例を実現するレジスタファイルポートアクセス回路を示すハードウエアブロック図。

【図７】本発明を実現するレジスタファイルシステムの一例を示すハードウエアブロック図。

【符号の説明】

ＢＯＶ，ＢＯＬ，ＴＯＬ，ＯＴＯＬ，ＢＯＲ，ＴＯＲ：ベースレジスタ

５０：スタティックレジスタ

６０：割り当てられていないアドレス空間

１４０，１６０：レジスタファイルポートアクセス回路

１４４：レジスタファイル

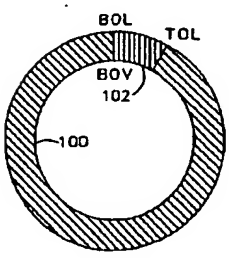
１４６：マルチプレクサ

１５０，１６４，１６６，１６８：コンパレータ

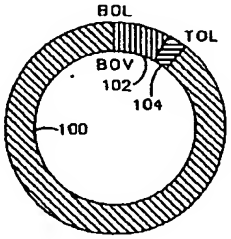
１５２：マルチプレクサ制御ライン

１５４，１７０，１７２：加算器

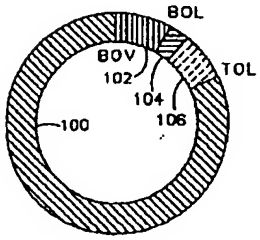
【図４Ａ】



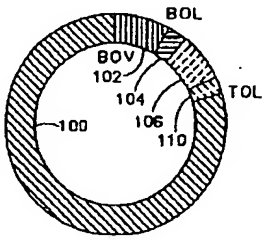
【図４Ｂ】



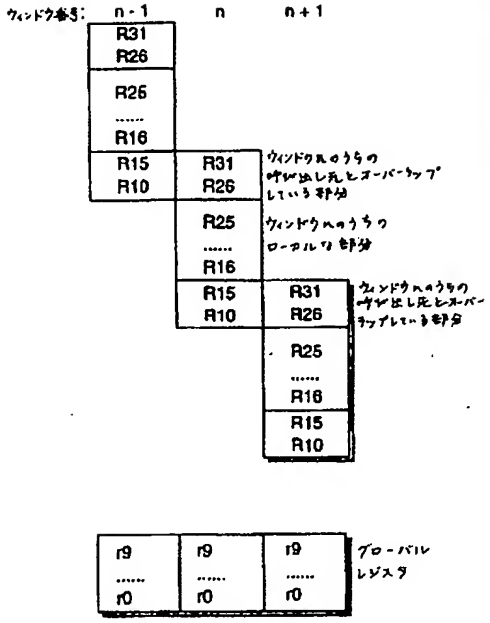
【図４Ｃ】



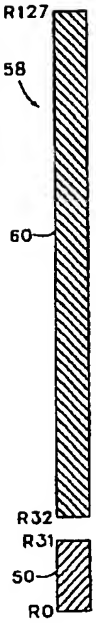
【図４Ｄ】



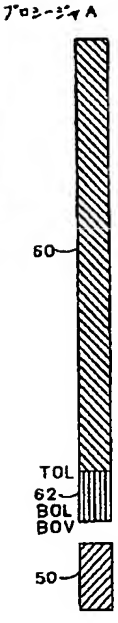
【図１】



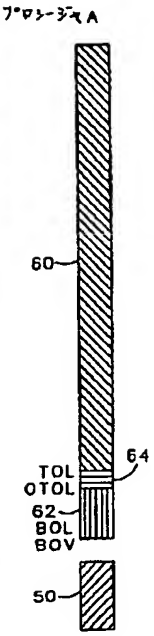
【図２】



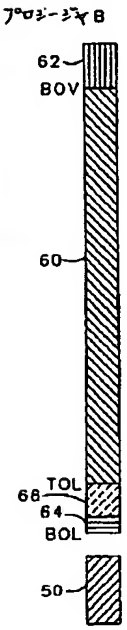
【図３Ａ】



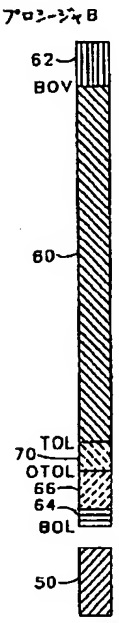
【図３Ｂ】



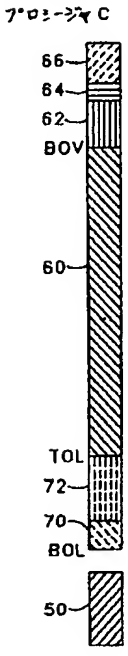
【図３Ｃ】



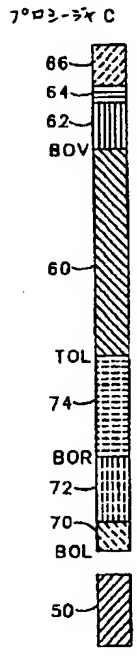
【図３Ｄ】



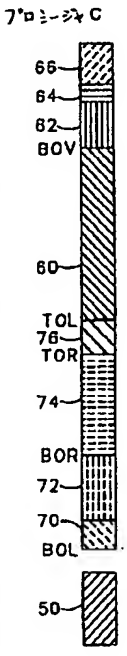
【図３Ｅ】



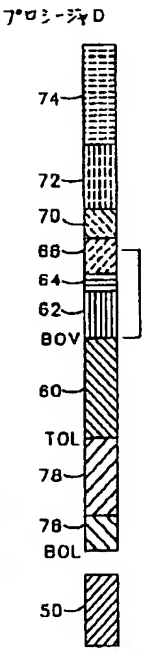
【図３Ｆ】



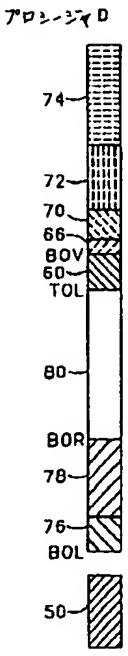
【図３Ｇ】



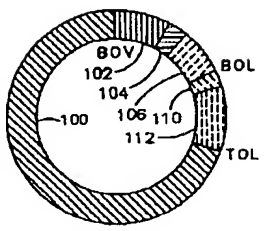
【図３Ｈ】



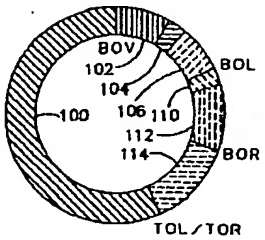
【図３Ｉ】



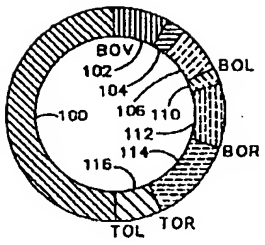
【図４Ｅ】



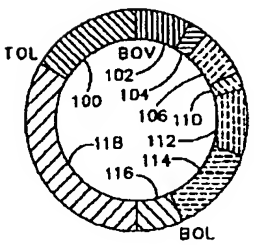
【図４Ｆ】



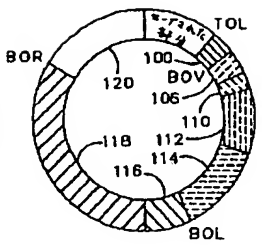
【図４Ｇ】



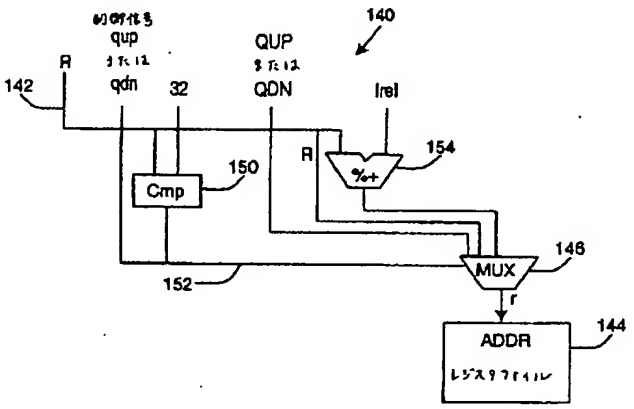
【図４Ｈ】



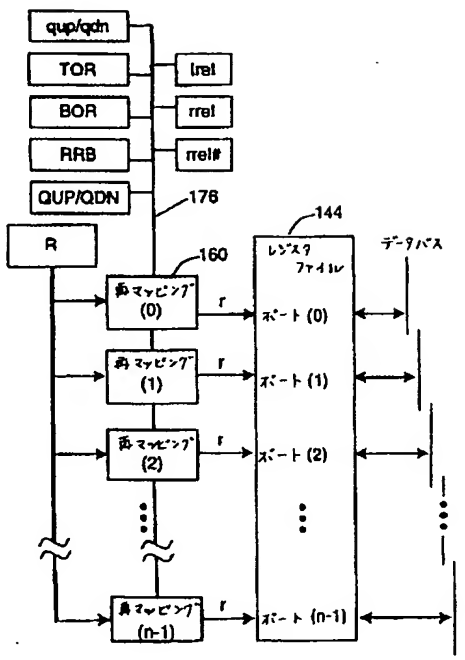
【図４Ｉ】



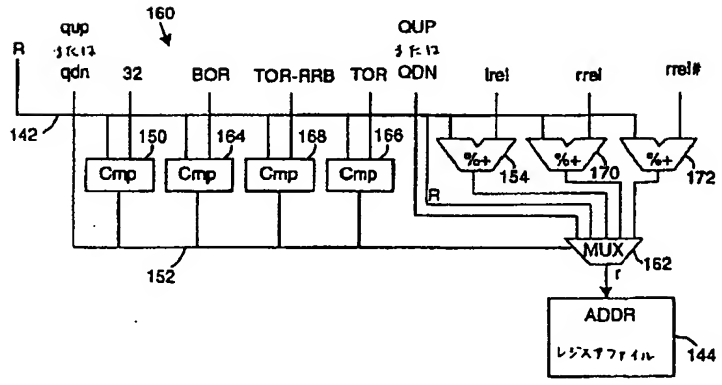
【図５】



【図７】



【図６】



フロントページの続き

(72)発明者　ロバート・エム・イングリッシュ
アメリカ合衆国カリフォルニア州メンロー・パーク、イースト・クリーク・プレイス　4

(72)発明者　ラジブ・グプタ
アメリカ合衆国カリフォルニア州ロス・アルトス、エコー・ドライブ　1052

(72)発明者　渡邊　坦
神奈川県川崎市麻生区王禅寺1099番地　株式会社日立製作所　システム開発研究所内

【公報種別】特許法第１７条の２の規定による補正の掲載
【部門区分】第６部門第３区分
【発行日】平成１４年７月１９日（２００２．７．１９）

【公開番号】特開平７－２８１８９７
【公開日】平成７年１０月２７日（１９９５．１０．２７）
【年通号数】公開特許公報７－２８１９
【出願番号】特願平７－１０８０５６
【国際特許分類第７版】
G06F 9/42 330
9/46 313
【ＦＩ】
G06F 9/42 330 R
9/46 313 B

【手続補正書】
【提出日】平成１４年３月１９日（２００２．３．１９）
【手続補正１】
【補正対象書類名】明細書
【補正対象項目名】特許請求の範囲
【補正方法】変更
【補正内容】
【特許請求の範囲】
【請求項１】コンパイラの介在なしに、デジタルコンピュータにおいてプロシージャにレジスタを動的に割り当てる方法であって、
複数のスタックレジスタを有する論理レジスタスタックを定義するステップと、
前記論理レジスタスタックを物理レジスタセットにマッピングするためのオフセットを定義するために、ローカルリロケーション項（ｌｒｅｌ）を初期化するステップと、
第１のスタックポインタ値（ＴＯＬ）を初期化することにより、第１のプロシージャによって指定される任意の数のスタックレジスタをローカルレジスタとして該第１のプロシージャに割り当てて、前記論理レジスタスタックにおいて該ローカルレジスタの範囲を区切るステップと、
第１の循環レジスタポインタ値（ＢＯＲ）および第２の循環レジスタポインタ値（ＴＯＲ）を、前記第１のスタックポインタ値（ＴＯＬ）に初期化するステップと、
前記第１のプロシージャによって循環レジスタとして指定されたレジスタの任意の数だけ、前記第２の循環レジスタポインタ値（ＴＯＲ）および前記第１のスタックポインタ値（ＴＯＬ）をインクリメントすることにより、レジスタを、循環レジスタとして前記第１のプロシージャに割り当てるステップと、
前記第１のプロシージャから復帰するに先立って、前記第２の循環レジスタポインタ値（ＴＯＲ）および前記第１のスタックポインタ値（ＴＯＬ）を、該第１のプロシージャの循環レジスタの数だけデクリメントすることにより、該循環レジスタの割り当てを解除するステップと、
前記第１のプロシージャの実行中のレジスタアクセス操作に関連して、前記ローカルリロケーション項（ｌｒｅｌ）に応答して物理アドレス（ｒ）を求めることにより、ローカルレジスタのそれぞれの論理レジスタアドレス（Ｒ）を、前記物理レジスタにマッピングするステップと、を含み、
前記物理レジスタにマッピングするステップは、
（ａ）前記論理レジスタアドレス（Ｒ）がスタティックレジスタの範囲内にあるならば、該論理レジスタアドレス（Ｒ）に等しくなるよう前記物理アドレスをセットし、
（ｂ）前記論理レジスタアドレス（Ｒ）が前記スタティックレジスタの範囲内になく、かつ該論理レジスタが循環レジスタでないならば、前記論理レジスタ（Ｒ）および前記ローカルリロケーション項（ｌｒｅｌ）を、前記任意の数のスタックレジスタを法とし、前記スタティックレジスタの範囲だけオフセットされたモジューロ加算を実行することで、前記物理アドレスをセットし、
（ｃ）前記論理レジスタアドレス（Ｒ）がスタティックレジスタの範囲内になく、かつ前記論理レジスタが循環レジスタならば、前記論理レジスタアドレス（Ｒ）が、前記ＴＯＲおよびＢＯＲによって定義される循環レジスタの範囲内にあるかどうかを判断し、
（ｃ－１）前記論理アドレスＲが前記循環レジスタの範囲内にあるならば、前記論理レジスタアドレス（Ｒ）および循環リロケーション項（ｒｒｅｌ）のモジューロ加算したものに前記物理アドレスをセットし、該循環リロケーション項（ｒｒｅｌ）は、循環レジスタベース値（ＲＲＢ）に前記ローカルリロケーション項（ｌｒｅｌ）を加算したものに等しく、

（ｃ－２）前記論理レジスタアドレス（Ｒ）が前記循環レジスタの範囲内にないならば、前記論理レジスタアドレス（Ｒ）および別の循環リロケーション項（＃ｒｒｅｌ）のモジューロ加算したものに前記物理アドレスをセットし、該別の循環リロケーション項（＃ｒｒｅｌ）は、前記循環レジスタベース値（ＲＲＢ）から前記循環レジスタの範囲を差し引いたものを前記ローカルリロケーション項（ｌｒｅｌ）に加算したものに等しい、プロシージャにレジスタを動的に割り当てる方法。

【請求項２】前記第１のスタックポインタ値（ＴＯＬ）をストアして、第２のスタックポインタ値（ＯＴＯＬ）を形成するステップと、

前記第１のスタックポインタ値（ＴＯＬ）をインクリメントすることにより、前記第１のプロシージャによって指定される任意の数の追加スタックレジスタを、パラメータ引き渡しレジスタとして該第１のプロシージャに割り当てて、該パラメータ引き渡しレジスタを含むようにするステップと、

呼び出されたプロシージャが参照するために、前記割り当てられたパラメータ引き渡しレジスタに少なくとも１つのパラメータをストアするステップと、をさらに含み、

前記少なくとも１つのパラメータをストアする前記ステップは、前記ローカルリロケーション項に応答して、前記パラメータ引き渡しレジスタを前記物理レジスタセットにマッピングすることを含む、請求項１に記載のプロシージャにレジスタを動的に割り当てる方法。

【請求項３】第２のプロシージャを呼び出すステップと、

前記第１のプロシージャのパラメータ引き渡しレジスタを有する初期ローカルレジスタスペースを前記第２のプロシージャに割り当て、該レジスタにストアされた前記少なくとも１つのパラメータを、メモリ参照なしに前記第２のプロシージャに利用可能なようにするステップと、

前記スタックポインタ値をインクリメントすることにより、前記第２のプロシージャによって指定される任意の数の追加のスタックレジスタを、ローカルレジスタとして該第２のプロシージャに割り当て、前記第１のプロシージャのローカルレジスタの内容をはじめにメモリにセーブすることなしに、該第２のプロシージャのローカルレジスタを含むようにするステップと、

前記第２のプロシージャから復帰する時、ローカルレジスタの数だけ前記スタックポインタ値をデクリメントすることによって、該ローカルレジスタの割り当てを解除し、これによって、ローカルレジスタの内容をセーブおよび復元することなく、該第２のプロシージャを呼び出し、またそこから復帰するステップと、

をさらに含む、請求項２に記載のプロシージャにレジスタを動的に割り当てる方法。

【請求項４】前記第２のプロシージャを呼び出す時、前記第１および第２のスタックポインタ値（ＴＯＬ、ＯＴＯＬ）をストアして、該第２のプロシージャから復帰する際に参照するためのストアされた値を形成するステップをさらに含み、

前記割り当てを解除するステップは、前記第１および第２のスタックポインタ値を前記ストアされた値にリセットすることを含む、請求項３に記載のプロシージャにレジスタを動的に割り当てる方法。

【請求項５】追加のパラメータレジスタを前記第１のプロシージャに割り当てるステップをさらに含み、

前記第１および第２のスタックポインタ値をストアする前記ステップは、前記第２のプロシージャからの復帰の際に参照するために、スタックポインタオフセット値を前記追加のパラメータレジスタにストアすることを含む、請求項４に記載のプロシージャにレジスタを動的に割り当てる方法。

【請求項６】前記割り当てを解除するステップは、前記追加のパラメータレジスタにストアされた前記スタックポインタオフセット値からスタックポインタ値を計算し、該計算された値に、前記スタックポインタ値をリセットすることを含む、請求項５に記載のプロシージャにレジスタを動的に割り当てる方法。

【請求項７】前記第２のプロシージャにおいて、前記第１のプロシージャのパラメータ引き渡しレジスタのうちの選択された一つに、計算された値をストアすることによって、該計算された値を前記第１のプロシージャに返すステップをさらに含む、請求項３に記載のプロシージャにレジスタを動的に割り当てる方法。

【請求項８】スタティックレジスタセットを提供し、前記第２のプロシージャにおいて、該スタティックレジスタの選択された一つに、計算された値をストアすることによって、該計算された値を前記第１のプロシージャに返すステップをさらに含み、

これにより、前記パラメータレジスタが、次のプロシージャ呼び出しに即座に利用可能なようにする、請求項３に記載のプロシージャにレジスタを動的に割り当てる方法。

【請求項９】レジスタファイルポートをアクセスする物理アドレスを提供するためのレジスタファイルポートアクセス装置であって、

現プロシージャから仮想アドレス（Ｒ）を受け取る入力手段と、

前記仮想アドレス（Ｒ）を予め決められた定数と比較して、該仮想アドレスがスタックレジスタのスタティックレジスタを示すかどうかを判断する第１のコンパレータ手段と、

前記仮想アドレスをローカルリロケーション項（ｌｒｅｌ）に加算して、第１の物理アドレスを形成する手段と、

前記仮想アドレスおよび前記第１の物理アドレス（ｒ）のうちの一つを選択して、該選択されたアドレスを前記レジスタファイルポートに接続する第１のマルチプレクサ手段と、

前記第１のマルチプレクサ手段に結合され、前記仮想アドレスがスタックレジスタを示すならば前記第１の物理アドレスを選択し、前記仮想アドレスがスタティックレジスタを示すならば該仮想アドレスを選択し、これにより、スタックレジスタ参照を、前記現プロシージャに割り当てられた物理レジスタアドレスに振り替える制御手段と、

前記仮想アドレス（Ｒ）を第１および第２の循環レジスタポインタ値（ＢＯＲ，ＴＯＲ）と比較し、該仮想アドレスが、前記現プロシージャに循環レジスタとして割り当てられたレジスタを示すかどうかを判断する第２のコンパレータ手段と、

前記仮想アドレス（Ｒ）を第１の循環リロケーション項（ｒｒｅｌ）に加算して、第１の物理アドレスを形成する手段と、

前記仮想アドレス（Ｒ）を第２の循環リロケーション項（ｒｒｅｌ＃）に加算して、第２の物理アドレスを形成する手段と、

前記第１および第２の物理アドレスのうちの一つを選択し、該選択されたアドレスを、前記レジスタファイルポートアドレス端子に結合する第２のマルチプレクサ手段と、

前記第２のマルチプレクサ手段を制御して、前記仮想アドレス（Ｒ）が、前記循環レジスタセット内の回り込みを示さなければ前記第１の物理アドレスを選択し、前記仮想アドレスが前記循環レジスタ内の回り込みを示すならば前記第２の物理アドレスを選択する制御手段と、を備え、

前記第１の循環リロケーション項（ｒｒｅｌ）は、前記ローカルリロケーション項（ｌｒｅｌ）に前記循環レジスタのベース値（ＲＲＢ）を加算したものに等しく、前記第２の循環リロケーション項（ｒｒｅｌ＃）は、前記循環レジスタベース値（ＲＲＢ）から前記循環レジスタセットのサイズを差し引いたものを、前記ローカルリロケーション項（ｌｒｅｌ）に加算したものに等しく、これにより、前記循環レジスタセット内の回り込みを調整する、

レジスタファイルポートアクセス装置。